URN (Paper): urn:nbn:de:gbv:ilm1-2014iwk-197:3

58th ILMENAU SCIENTIFIC COLLOQUIUM
Technische Universität Ilmenau, 08 – 12 September 2014 URN: urn:nbn:de:gbv:ilm1-2014iwk:3

SICALT – SPECTRAL IMAGE CORRECTION AND ANALYSIS TOOL

Hans-Christian Schwannecke, Richard Fütterer, Maik Rosenberger, Gerhard Linß

Ilmenau University of Technology, Faculty of Mechanical Engineering, Department of Quality Assurance and Industrial Image Processing, Gustav Kirchhoff Platz 2, 98693 Ilmenau, Germany

ABSTRACT

Enhancing, displaying and analyzing multi and hyper spectral data is the key aspect of this paper. For this purpose the "Spectral Image Correction an Analysis Tool" – SICALT was developed in our department under the use of the MatLab language and the model view presenter software design pattern. With SICALT we want to have a tool at hand that can transform data from multi or hyper spectral imagers into a form, usable by standard quality assurance image processing tools. We demonstrate the functionality on a printed circuit board by the segmentation of different materials.

Index Terms **–** spectral imaging, quality assurance, industrial measurement

1. INTRODUCTION

Multi and hyper spectral imaging is challenging in many ways. The following paper deals with the problem of the displaying, enhancing and analyzing of pictures from multi and hyper spectral imaging sources. Some other tools like HIAT [1] or MultiSpec [2] exist, but these tools are developed for the use with spectral data from satellites or planes like LANDSAT MSS or EO-1 Hyperion. In quality assurance and industrial measurement are other needs to consider. An example is the estimation of an exact location of the edge between two materials. Therefore the "Spectral Image Correction an Analysis Tool" – SICALT was developed in our department.

The paper structure follows the development process. After determining the requirements for the application, the architecture will be described. In addition, some remarks about the used software design pattern and their application in the software architecture will be made. The next section contains an overall description of the main application features. Finally, some examples will demonstrate the software functionality.

2. REQUIREMENTS

A use case diagram is a type of diagrams specified by the unified modelling language (UML) [3], used for graphical presentation and documentation of assignation of requirements and various operation cases or members. Figure 1 shows the demanded requirements in such an use case diagram. As a first task arose the inspection of the images from each channel. So, a possibility for the selection and the displaying of each spectral channel has to be given. Continuing from these single channels, the calculation and the displaying of a RGB composite image was desired. Another requirement was the optional change of different view modes. This should include the three modes: the raw data, the absorption spectrum and the reflection / transmission spectrum. Furthermore, the selection and displaying of several spectral

signatures should be implemented into the functionality of the tool. These signatures should enable the search within the spectral data, in order to enhance the boundary line of pixel groups with a different specified property, in other words the edges of material boarders. In addition, the tool should provide the supervised and unsupervised spectral unmixing. Finally, SICALT should come with a simply and easily adaptable user interface as well as the option for the simple extension of current features.

The last two requirements led to the use of the MatLab language, as MatLab offers a broad basis for the desired implementation with its huge library of ready to use algorithms and the build in graphical user interface (GUI) designer.

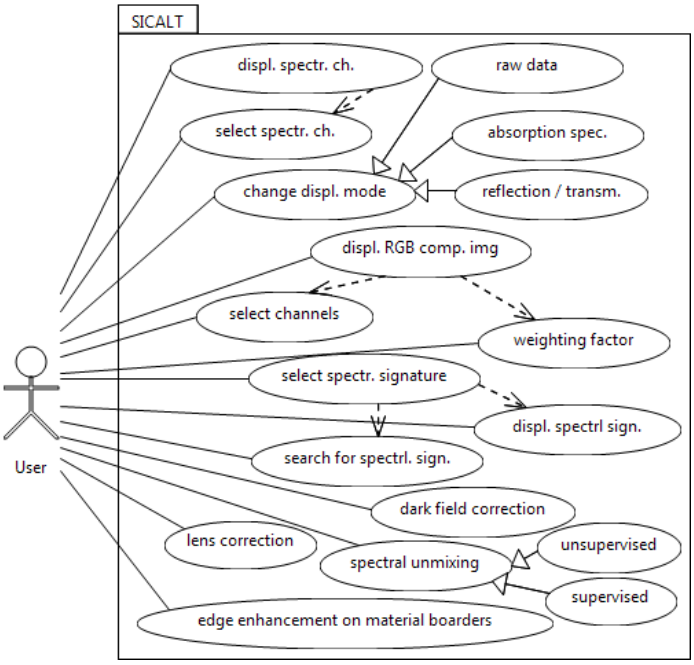

Figure 1: SICALT - Use Cases

3. ARCHITECTURE

A standard software design pattern was used for an easily adaptable piece of software. One specific pattern called model-view-controller (MVC) [4] is suitable for software with a graphical user interface. The aim of this pattern is the distribution of responsibilities to single modules of the software. Figure 2 shows the basic building blocks of the MVC pattern. The model represents the user data, the view stands for the GUI and the controller manages the interactions between view and model. Different types of communication occur in the MVC pattern. By accessing the model data the view displays the state of the model. On the other hand the model notifies about changes of its state, so the view will be kept to the most recent state. While this is a direct connection between view and model, the other direction is always executed over the controller. If the user triggers an event in the view, this event has to be executed by the controller. Only the controller is able to alter the model. After the alteration of the model is accomplished the view is notified by the model and updates itself.

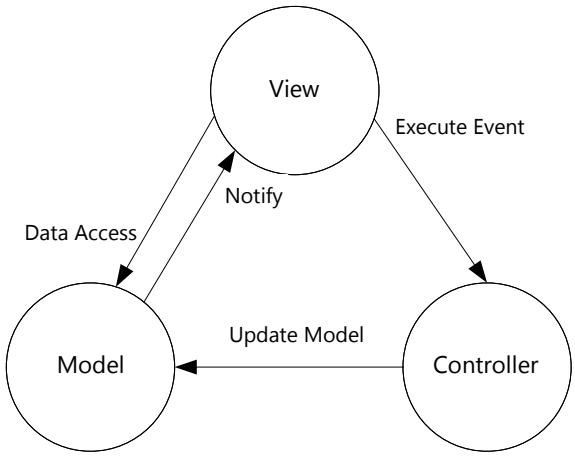

Figure 2: simple model view controller scheme

With the evolution of the GUI several variations of this pattern have been developed. These are for example, the model-view-view-model pattern, the presentation-model pattern and the

model-view-presenter (MVP) pattern [5, 6]. Our interest lies in the last one. This pattern takes new developments of the graphical user interfaces into account. Within the evolution of GUIs several functionalities, like the handling of the raw user interface events, have been transferred into the view objects, which changed the role of the controller into a presenter. Also the function of the model has been slightly altered. While in the MVC pattern the model must contain logic for accessing the user interface, this is not the case in MVP. The resulting MVP pattern as postulated in [7] is shown in Figure 3. As one can see, the presenter is the sole component communicating with the user interface and the model. The communication of the view is stripped down to the notification mechanism. Both the MVP and the MVC pattern used an observer mechanism to connect multiple views with the same model. An observer does as the name suggests. It is a software class that tracks changes on the model class and notifies or updates any registered observer. As this application is not yet designed to have multiple views, the connection between model and view is a one on one connection. So we can omit the observer.

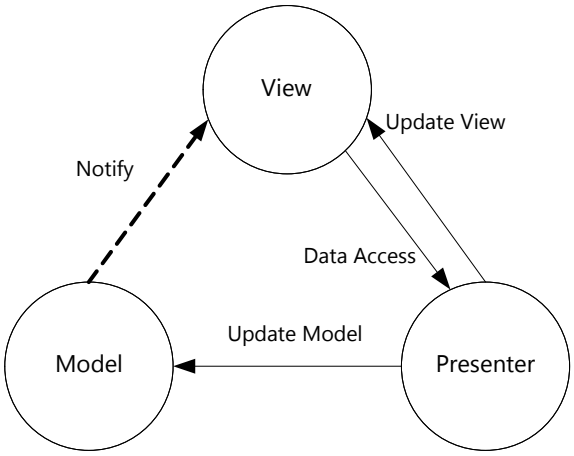

Figure 3 : basic model-view-presenter structure

Figure 4 shows a simplified UML class diagram of the static structure of the SICALT application. The lambda stack (*LambdaStack*) is the main class for the given spectral data and represents the data model in the MVP pattern. Here is a list of the different spectral channels stored. In this class are integrated only different access and data manipulation functions. All the functional operations are kept in the project class (*Project*). This class calculates and keeps the RGB composite image (*rgbImage*). This image is constructed from the lambda stack under the use of a mix down matrix (*mixdownMatrix*) given by the view. The selected pixel spectrum (*selectedPixSpec*) and a list of spectral signatures (*specSigntList*) for the later image segmentation are also located at the project class as well as the functions for image

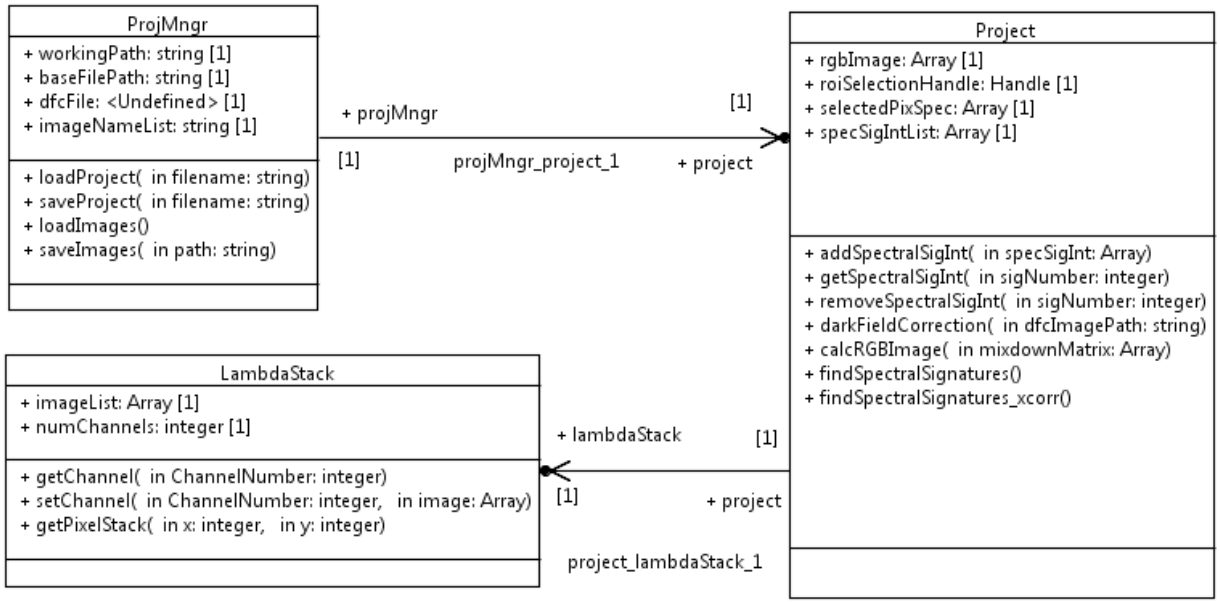

Figure 4: SICALT simplified static class structure

analysis and enhancement. Furthermore, a GUI handle is held that represents a selected region of interest (*roiSelectionHandle*).

The third shown class in Figure 4 is the project manager (*ProjMngr*) class. Here are kept all variables and functions, which are related to the file management. Also, this is the main access point for the GUI functionality. So the project manager creates together with the project class the presenter of the MVP pattern.

The view is only represented by MatLab GUI objects. The connection between view and presenter is accomplished by callback routines. In the case of a user event this routine is called from the view and executes the required behavior with the help of the presenter on the model.

4. APPLICATION

Figure 5 shows the main application window, where the different sections of the application are presented. The image display section is at the upper left. The single spectral channels or an assembled RGB image are shown here, respectively. The channel selection at the left of the image display section enables the user to choose between the spectral channels. Here, the user can also mark single channels for the assembly integration into the RGB image. While the original layout intents the use of a twelve channel multi spectral filter wheel camera [8], more channels are easily possible. The selection of the display modes is located in the lower right. Here, the user can choose between displaying the raw data, reflection and transmission data or absorption data of the spectral images. The lower middle section shows the display of the spectral signature. A "spectral fingerprint", which consists of the chosen pixel of the image in every spectral channel, will be shown by mouse click at the displayed image. Up to nine of these fingerprints can be added for future reference. With these fingerprints the image segmentation can be realized resulting in either a binary image with only the selected material or a gray scale image with enhanced edges. In the first case the binary image will be calculated with a confidence interval consisting of a given tolerance for each spectral channel, while for the second case the grey scale image is the result of a cross correlation between the spectral fingerprint and the entire lambda stack. On these processed images further measurements can be performed. In the lower section on the right, the interface for the unsupervised spectral unmixing is located. Currently, it is an unimplemented placeholder and will be filled with functionality in future versions. Three menu items are at the top of the

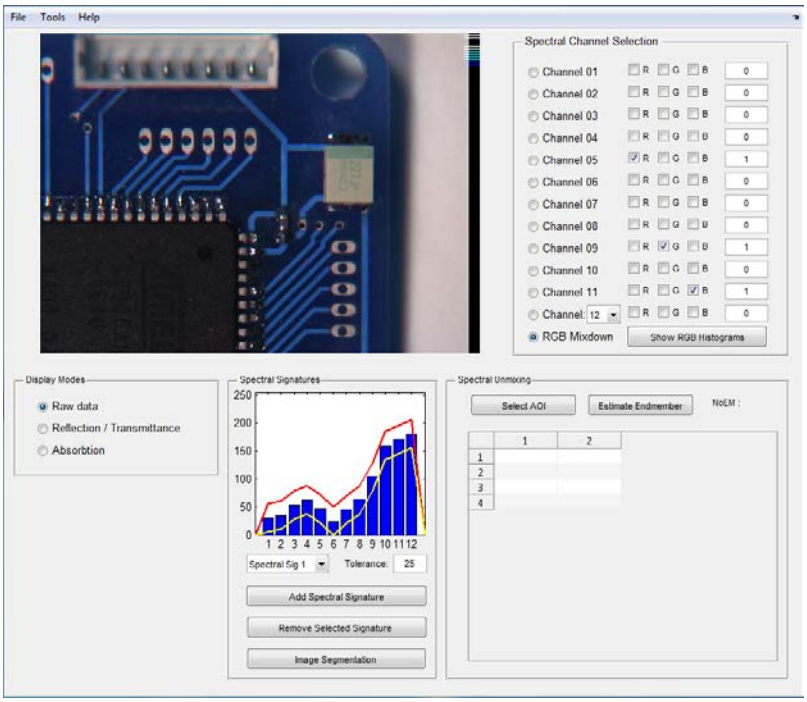

Figure 5: SICALT, main application window.

program window positioned. Under the file menu the actions for loading and saving projects and image files are accommodated. Less frequently used functions are located in the tool menu. Currently, these are dark field correction and circle center detection for the future function of lens correction. Planned is the application documentation contained in the help menu item.

5. EXAMPLE

As an example, the inspection of printed circuit boards (PCB) was investigated. In Figure 5 four images of a PCB part are shown. The first on the left side is an RGB composite image, while the others are images of different spectral segmentations. These images are a result of picking different spectral fingerprints from the image display section in the main application window (Figure 4). The second image at the left side presents the backplane of a PCB board, followed further to the right by the resulting selection of a spectral fingerprint from the conducting tracks of the PCB board, and the last image, furthermost to the right, gives the view of a selected single component.

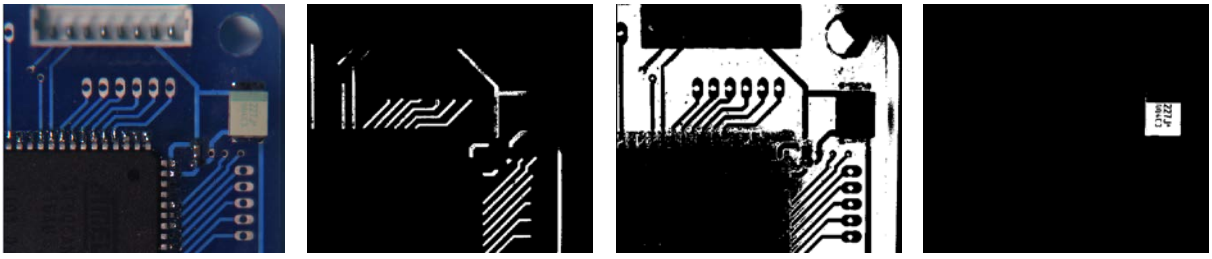

Figure 6: (left to right) RGB composite image; conductive tracks; backplane; single component

The segmentation was executed by using a confidence interval for every spectral channel in the fingerprint. The process examines for every pixel in every channel whether the intensity value lies within the confidence interval. Only if every channel fulfills the condition, the pixel in the resulting image is set to one. This procedure creates effectively the binary image as seen above. Although this is an easy calculation, the results may not be satisfying. Problems occur by an insufficient lighting of the scene and or by a missing filter or lens correction. E.g. a missing filter correction does lead to a shift of pixels between the single spectral channels and distorts the results.

For the purpose of measurement another approach was tested, whose resulting images are shown in Figure 6. Here, the segmentation was done using a cross correlation approach. This is a standard method for estimating the degree to which two series are related [9]. In contrast to the above approach, the results are grey scale images where higher pixel intensity corresponds to a greater correlation value between the spectral fingerprint and the spectral channel intensity values at the current position.

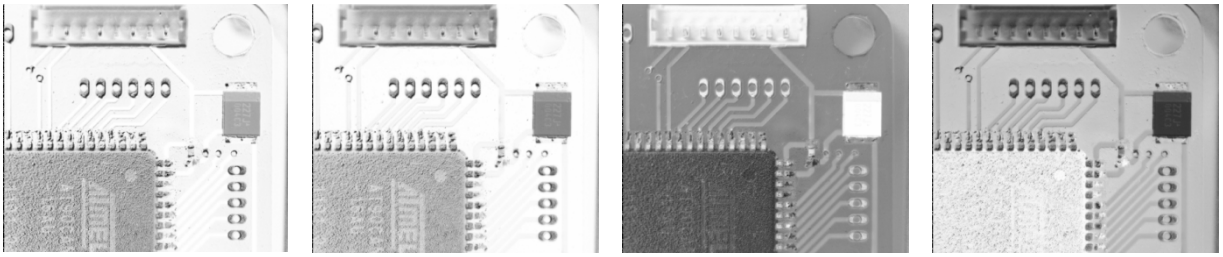

Figure 7 : correlation results(l. t. r.), conductive tracks, backplane, component 1, component 2

The goal was to create an image fit for the use as input for state of the art of industrial image processing. With this we can use edge detection algorithms and measure the physical dimensions of certain components.

6. CONCLUSION

SICALT provides most of the features postulated in the beginning. While some of them, like spectral unmixing are still in experimental status, others are already usable. The tool was developed using the MatLab language and with the use of the model-view-presenter software design pattern, which leads to an easily adaptable architecture and thus to a tool with a high reusability value. The functionality of the tool was beside others tested on a printed circuit board, showing two different types of segmentation algorithms (confidence interval and cross correlation). In future it is planned to expand the options for the segmentation, to integrate the functions for the unsuperwised spectral unmixing and to add the functionality for a lens correction algorithm.

REFERENCES

[1] Arzuaga-Cruz E., et. al., "A MATLAB Toolbox for Hyperspectral Image Analysis." Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 IEEE International

[2] Biehl L., Landgrebe D., "MultiSpec – a tool for multispectral –hyperspectral image data analysis", Computers & Geosciences, Volume 28, Issue 10, December 2002.

[3] Object Management Group, UML 2 Specification, [Online], http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/

[4] Krasner Glenn E., Pope Stephen T., „A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80", Journal of Object-Oriented Programming, Volume 1 Issue 3, Aug./Sept. 1988

[5] Shelest, Alex, „Model View Controller, Model View Presenter, and Model View ViewModel Design Patterns", http://www.codeproject.com/Articles/42830/Model-View-Controller-Model-View-Presenter-and-Mod,

[6] M. Fowler, GUI Architectures,[Online] http://martinfowler.com/eaaDev/uiArchs.html

[7] Bower, Andy; McGlashan, Blair; "TWISTING THE TRIAD, The evolution of the Dolphin Smalltalk MVP application framework."

[8] Preissler M., Rosenberger M., Linss G., "Smart Multispectral Imager", IEEE Photonics Conference 2012, Burlingham, CA

[9] P. Bourke; „Cross Correlation". [Online]. http://paulbourke.net/miscellaneous/correlate/

CONTACTS

Dipl.-Inf. Hans-Christian Schwannecke hans-christian.schwannecke@tu-ilmenau.de
Prof. Dr.-Ing. habil. G. Linß gerhard.linss@tu-ilmenau.de